EBook Gratis

APRENDIZAJE hadoop

Free unaffiliated eBook created from
Stack Overflow contributors.

#hadoop

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: hadoop

It is an unofficial and free hadoop ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official hadoop.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con hadoop

Observaciones

¿Qué es Apache Hadoop?

La biblioteca de software Apache Hadoop es un marco que permite el procesamiento distribuido de grandes conjuntos de datos en grupos de computadoras utilizando modelos de programación simples. Está diseñado para escalar desde servidores individuales a miles de máquinas, cada una ofrece computación y almacenamiento locales. En lugar de confiar en el hardware para ofrecer alta disponibilidad, la biblioteca en sí está diseñada para detectar y manejar fallas en la capa de aplicación, por lo que ofrece un servicio de alta disponibilidad sobre un grupo de computadoras, cada una de las cuales puede ser propensa a fallas.

Apache Hadoop incluye estos módulos:

- **Hadoop Common** : las utilidades comunes que admiten los otros módulos de Hadoop.
- **Sistema de archivos distribuidos de Hadoop (HDFS)** : un sistema de archivos distribuidos que proporciona acceso de alto rendimiento a los datos de la aplicación.
- **Hadoop YARN** : un marco para la planificación de tareas y la gestión de recursos de clúster.
- **Hadoop MapReduce** : un sistema basado en YARN para el procesamiento paralelo de grandes conjuntos de datos.

Referencia:

Apache Hadoop

Versiones

Versión	Notas de lanzamiento	Fecha de lanzamiento
3.0.0-alfa1		2016-08-30
2.7.3	Haga clic aquí - 2.7.3	2016-01-25
2.6.4	Haga clic aquí - 2.6.4	2016-02-11
2.7.2	Haga clic aquí - 2.7.2	2016-01-25
2.6.3	Haga clic aquí - 2.6.3	2015-12-17
2.6.2	Haga clic aquí - 2.6.2	2015-10-28

Versión	Notas de lanzamiento	Fecha de lanzamiento
2.7.1	Haga clic aquí - 2.7.1	2015-07-06

Examples

Instalación o configuración en Linux

Un procedimiento de configuración de clúster pseudo-distribuido

Prerrequisitos

- Instale JDK1.7 y establezca la variable de entorno JAVA_HOME.
- Crea un nuevo usuario como "hadoop".

  `useradd hadoop`

- Configuración de inicio de sesión SSH sin contraseña en su propia cuenta

```
su - hadoop
ssh-keygen
<< Press ENTER for all prompts >>
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
```

- Verificar realizando `ssh localhost`
- Deshabilite IPV6 editando `/etc/sysctl.conf` con lo siguiente:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

- Compruebe que utilizando `cat /proc/sys/net/ipv6/conf/all/disable_ipv6`

  (debe devolver 1)

Instalacion y configuracion:

- Descargue la versión requerida de Hadoop desde los archivos de Apache usando el comando `wget` .

```
cd /opt/hadoop/
wget http:/addresstoarchive/hadoop-2.x.x/xxxxx.gz
tar -xvf hadoop-2.x.x.gz
mv hadoop-2.x.x.gz hadoop
(or)

ln -s hadoop-2.x.x.gz hadoop
chown -R hadoop:hadoop hadoop
```

- Actualice `.bashrc` / `.kshrc` función de su shell con las siguientes variables de entorno

```
export HADOOP_PREFIX=/opt/hadoop/hadoop
export HADOOP_CONF_DIR=$HADOOP_PREFIX/etc/hadoop
export JAVA_HOME=/java/home/path
export PATH=$PATH:$HADOOP_PREFIX/bin:$HADOOP_PREFIX/sbin:$JAVA_HOME/bin
```

- En el `$HADOOP_HOME/etc/hadoop` edite debajo de los archivos

  - core-site.xml

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:8020</value>
    </property>
</configuration>
```

  - mapred-site.xml

    Crea `mapred-site.xml` desde su plantilla

    `cp mapred-site.xml.template mapred-site.xml`

```
 <configuration>
     <property>
         <name>mapreduce.framework.name</name>
         <value>yarn</value>
     </property>
 </configuration>
```

  - hilo-sitio.xml

```
<configuration>
    <property>
        <name>yarn.resourcemanager.hostname</name>
        <value>localhost</value>
    </property>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

  - hdfs-site.xml

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:///home/hadoop/hdfs/namenode</value>
    </property>
```

```
    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:///home/hadoop/hdfs/datanode</value>
    </property>
</configuration>
```

Cree la carpeta principal para almacenar los datos de hadoop

```
mkdir -p /home/hadoop/hdfs
```

- Formato NameNode (limpia el directorio y crea los meta archivos necesarios)

```
hdfs namenode -format
```

- Inicia todos los servicios:

```
start-dfs.sh && start-yarn.sh
mr-jobhistory-server.sh start historyserver
```

En su lugar, use start-all.sh (en desuso).

- Compruebe todos los procesos en ejecución de Java

```
jps
```

- Interfaz Web de Namenode: http: // localhost: 50070 /
- Interfaz web del administrador de recursos: http: // localhost: 8088 /
- Para detener demonios (servicios):

```
stop-dfs.sh && stop-yarn.sh
mr-jobhistory-daemon.sh stop historyserver
```

En su lugar, use stop-all.sh (en desuso).

Instalación de Hadoop en ubuntu

Creando Usuario Hadoop:

```
sudo addgroup hadoop
```

Añadiendo un usuario:

```
sudo adduser --ingroup hadoop hduser001
```

```
sandeep-001@ubuntu-001:~$ java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) Client VM (build 24.51-b03, mixed mode)
sandeep-001@ubuntu-001:~$ javac -version
javac 1.7.0_51
sandeep-001@ubuntu-001:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1001) ...
Done.
sandeep-001@ubuntu-001:~$ sudo adduser --ingroup hadoop hduser001
Adding user `hduser001' ...
Adding new user `hduser001' (1001) with group `hadoop' ...
Creating home directory `/home/hduser001' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser001
Enter the new value, or press ENTER for the default
        Full Name []: Sandeep Chatterjee
        Room Number []: 001
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
sandeep-001@ubuntu-001:~$
```

Configurando SSH:

```
su -hduser001
ssh-keygen -t rsa -P ""
cat .ssh/id_rsa.pub >> .ssh/authorized_keys
```

Nota : Si obtiene errores [*bash: .ssh / authorized_keys: No existe dicho archivo o directorio*] mientras escribe la clave autorizada. Compruebe aquí .

```
sandeep-001@ubuntu-001:~$ su - hduser001
Password:
hduser001@ubuntu-001:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser001/.ssh/id_rsa):
Created directory '/home/hduser001/.ssh'.
Your identification has been saved in /home/hduser001/.ssh/id_rsa.
Your public key has been saved in /home/hduser001/.ssh/id_rsa.pub.
The key fingerprint is:
de:58:56:43:7c:f9:f3:2e:85:73:db:58:59:d3:48:05 hduser001@ubuntu-001
The key's randomart image is:
+--[ RSA 2048]----+
|           .. E+.|
|           .. +  |
|            oo o.|
|           . ..o+|
|        S o    .*|
|       . =    o.=|
|        o .    Bo|
|              o.o|
|               . |
+-----------------+
hduser001@ubuntu-001:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
hduser001@ubuntu-001:~$
```


```
sandeep-001@ubuntu-001:~$ su - hduser001
Password:
hduser001@ubuntu-001:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser001/.ssh/id_rsa):
Created directory '/home/hduser001/.ssh'.
Your identification has been saved in /home/hduser001/.ssh/id_rsa.
Your public key has been saved in /home/hduser001/.ssh/id_rsa.pub.
The key fingerprint is:
de:58:56:43:7c:f9:f3:2e:85:73:db:58:59:d3:48:05 hduser001@ubuntu-001
The key's randomart image is:
+--[ RSA 2048]----+
|           .. E+.|
|           .. +  |
|            oo o.|
|           . ..o+|
|        S o    .*|
|       . =    o.=|
|        o .    Bo|
|              o.o|
|               . |
+-----------------+
hduser001@ubuntu-001:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
hduser001@ubuntu-001:~$
```

```
hduser001@ubuntu-001:~$ pwd
/home/hduser001
hduser001@ubuntu-001:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is a5:7c:22:56:dc:fa:1c:14:1e:66:0f:9e:ec:bb:b3:52.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.3 LTS (GNU/Linux 3.8.0-29-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Feb  6 18:29:41 IST 2014

  System load:  0.01              Processes:           74
  Usage of /:   6.7% of 18.32GB   Users logged in:     1
  Memory usage: 18%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  Graph this data and manage this system at https://landscape.canonical.com/


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hduser001@ubuntu-001:~$
```

Añadir usuario de hadoop a la lista de sudoers:

```
sudo adduser hduser001 sudo
```

```
sandeep-001@ubuntu-001:~$ sudo adduser hduser001 sudo
Adding user `hduser001' to group `sudo' ...
Adding user hduser001 to group sudo
Done.
sandeep-001@ubuntu-001:~$
```

Deshabilitando IPv6:

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
"/etc/sysctl.conf" 64L, 2205C written
sandeep-001@ubuntu-001:~$
```

```
sandeep-001@ubuntu-001:~$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
1
```

Instalación de Hadoop:

```
sudo add-apt-repository ppa:hadoop-ubuntu/stable
sudo apt-get install hadoop
```

```
sandeep-001@ubuntu-001:~$ sudo add-apt-repository ppa:hadoop-ubuntu/stable
[sudo] password for sandeep-001:
You are about to add the following PPA to your system:
 Hadoop Stable packages

These packages are based on Apache Bigtop with appropriate patches to enable nat
ive integration on Ubuntu Oneiric onwards and for ARM based archictectures.

Please report bugs here - https://bugs.launchpad.net/hadoop-ubuntu-packages/+fil
ebug
 More info: https://launchpad.net/~hadoop-ubuntu/+archive/stable
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmpo31xDq/secring.gpg' created
gpg: keyring `/tmp/tmpo31xDq/pubring.gpg' created
gpg: requesting key 84FBAFF0 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpo31xDq/trustdb.gpg: trustdb created
gpg: key 84FBAFF0: public key "Launchpad PPA for Hadoop Ubuntu Packagers" import
ed
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
sandeep-001@ubuntu-001:~$
```

```
sandeep-001@ubuntu-001:~$ sudo apt-get install hadoop
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bigtop-utils hadoop-native libsnappy1
The following NEW packages will be installed:
  bigtop-utils hadoop hadoop-native libsnappy1
0 upgraded, 4 newly installed, 0 to remove and 2 not upgraded.
Need to get 30.8 MB of archives.
After this operation, 34.5 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/universe libsnappy1 i386 1.0.
4-1build1 [13.0 kB]
Get:2 http://ppa.launchpad.net/hadoop-ubuntu/stable/ubuntu/ precise/main bigtop-
utils all 0.3.0~hadoop2 [6,766 B]
Get:3 http://ppa.launchpad.net/hadoop-ubuntu/stable/ubuntu/ precise/main hadoop
all 1.0.2-0ubuntu1~hadoop1 [30.8 MB]
5% [3 hadoop 1,589 kB/30.8 MB 5%]                              147 kB/s 3min 18s
```

Visión general de Hadoop y HDFS

Hadoop es un marco de software de código abierto para almacenamiento y procesamiento a gran escala de conjuntos de datos en un entorno informático distribuido. Está patrocinado por la Apache Software Foundation. Está diseñado para escalar desde servidores individuales a miles de máquinas, cada una ofrece computación y almacenamiento locales.

Historia

- Hadoop fue creado por Doug Cutting y Mike Cafarella en 2005.
- Cutting, que trabajaba en Yahoo! en ese momento, lo nombró por el juguete del elefante de su hijo.
- Originalmente fue desarrollado para soportar la distribución del proyecto del motor de búsqueda.

Los principales módulos de hadoop.

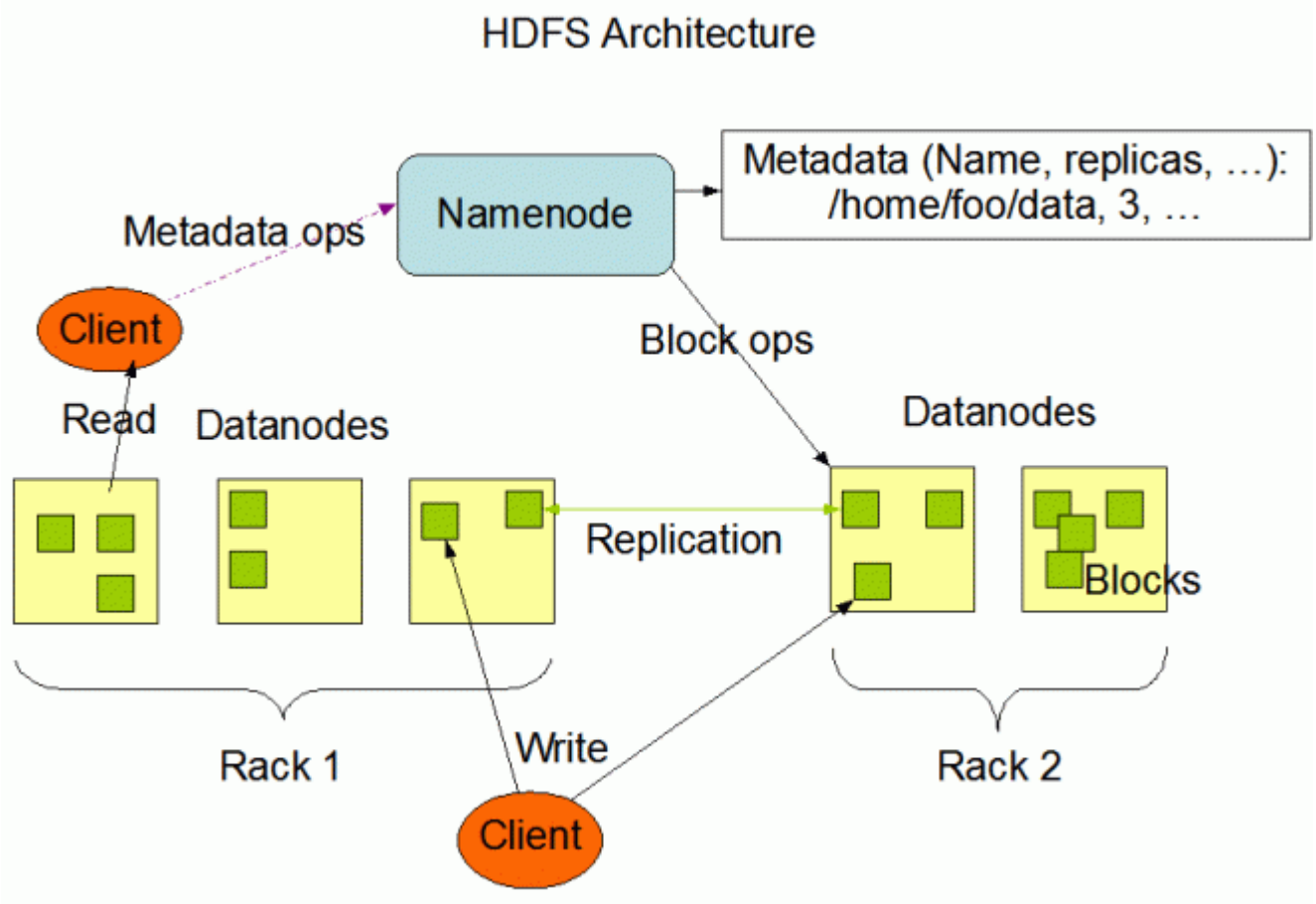
Sistema de archivos distribuidos de Hadoop (HDFS): un sistema de archivos distribuidos que proporciona acceso de alto rendimiento a los datos de la aplicación. Hadoop MapReduce: un marco de software para el procesamiento distribuido de grandes conjuntos de datos en clusters de cómputo.

Características básicas del sistema de archivos Hadoop

Muy tolerante a fallos. Alto rendimiento. Adecuado para aplicaciones con grandes conjuntos de datos. Puede ser construido de hardware de los productos básicos.

Namenode y Datanodes

Arquitectura maestro / esclavo. El clúster HDFS consta de un solo Namenode, un servidor maestro que administra el espacio de nombres del sistema de archivos y regula el acceso de los clientes a los archivos. Los DataNodes administran el almacenamiento adjunto a los nodos en los que se ejecutan. HDFS expone un espacio de nombres del sistema de archivos y permite que los datos del usuario se almacenen en archivos. Un archivo se divide en uno o más bloques y el conjunto de bloques se almacena en DataNodes. DataNodes: sirve para leer, escribir solicitudes, realizar la creación, eliminación y replicación de bloques siguiendo instrucciones de Namenode.



HDFS está diseñado para almacenar archivos muy grandes a través de máquinas en un grupo grande. Cada archivo es una secuencia de bloques. Todos los bloques en el archivo, excepto el último, son del mismo tamaño. Los bloques se replican para tolerancia a fallos. El Namenode recibe un Heartbeat y un BlockReport de cada DataNode en el clúster. BlockReport contiene todos los bloques en un Datanode.

Comandos de Shell Hadoop

Comandos comunes utilizados: -

ls Uso: **hadoop fs –ls Ruta** (dir / archivo ruta a lista). Uso del **gato** : **hadoop fs -cat PathOfFileToView**

```
harinder@harinder-laptop: ~
harinder@harinder-laptop:~$ hadoop fs -cat testData/Test.txt
Hi this is just a plain text file.

Thanks


harinder@harinder-laptop:~$
```

Enlace para los comandos de shell hadoop: - https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

Lea Empezando con hadoop en línea: https://riptutorial.com/es/hadoop/topic/926/empezando-con-hadoop

Capítulo 2: ¿Qué es HDFS?

Observaciones

Una buena explicación de HDFS y cómo funciona.

La sintaxis debe contener los comandos que pueden usarse en HDFS.

Examples

HDFS - Sistema de archivos distribuidos de Hadoop

Hadoop Distributed File System (HDFS) es un sistema de archivos basado en Java que proporciona almacenamiento de datos escalable y confiable que está diseñado para abarcar grandes grupos de servidores de productos básicos. HDFS, MapReduce y YARN forman el núcleo de Apache ™ Hadoop®.

HDFS está diseñado para ser altamente tolerante a fallos, lo que se logra al guardar varias copias (3 de manera predeterminada) de un bloque de datos dado en múltiples nodos.

Buscando archivos en HDFS

Para buscar un archivo en el sistema de archivos Hadoop Distributed:

```
hdfs dfs -ls -R / | grep [search_term]
```

En el comando anterior,

`-ls` es para listar archivos

`-R` es para recursivo (iterar a través de subdirectorios)

`/` significa desde el directorio raíz

`|` para canalizar la salida del primer comando al segundo

comando `grep` para extraer cadenas coincidentes

`[search_term]` nombre de archivo `[search_term]` que se buscará en la lista de todos los archivos en el sistema de archivos hadoop.

Alternativamente, el siguiente comando también se puede usar para encontrar y aplicar algunas expresiones:

```
hadoop fs -find / -name test -print
```

Busca todos los archivos que coinciden con la expresión especificada y les aplica las acciones

seleccionadas. Si no se especifica ninguna ruta, el directorio de trabajo actual se establece de forma predeterminada. Si no se especifica ninguna expresión, el valor predeterminado es -print.

Se reconocen las siguientes expresiones primarias:

- `name pattern`
- `iname pattern`

Se evalúa como verdadero si el nombre base del archivo coincide con el patrón utilizando el sistema estándar de archivos del globo. Si se usa -iname, la coincidencia no distingue entre mayúsculas y minúsculas.

- `print`
- `print0Always`

Evalúa a la verdad. Hace que la ruta de acceso actual se escriba en la salida estándar. Si se `-print0` expresión `-print0` se agrega un carácter ASCII NULL.

Se reconocen los siguientes operadores:

```
expression -a expression
expression -and expression
expression expression
```

Bloques y Splits HDFS

1. **Tamaño de bloque y bloques en HDFS** : HDFS tiene el concepto de almacenar datos en bloques cada vez que se carga un archivo. Los bloques son las particiones físicas de datos en HDFS (o en cualquier otro sistema de archivos, para el caso).

   Cada vez que se carga un archivo en el HDFS, se divide físicamente (sí, el archivo se divide) en diferentes partes conocidas como bloques. El número de bloques depende del valor de `dfs.block.size` en `hdfs-site.xml`

   Idealmente, el tamaño del bloque se establece en un valor alto, como 64/128/256 MB (en comparación con 4KB en FS normal). El valor de tamaño de bloque predeterminado en la mayoría de las distribuciones de Hadoop 2.x es de 128 MB. La razón para un tamaño de bloque más alto es porque Hadoop está hecho para tratar PetaBytes de datos con cada archivo que van desde unos pocos cientos de MegaBytes hasta el orden de TeraBytes.

   Digamos, por ejemplo, que tiene un archivo de tamaño de 1024 MB. Si su tamaño de bloque es de 128 MB, obtendrá 8 bloques de 128 MB cada uno. Esto significa que su namenode necesitará almacenar metadatos de `8 x 3 = 24` archivos (3 es el factor de replicación).

   Considere el mismo escenario con un tamaño de bloque de 4 KBs. El resultado será `1GB / 4KB = 250000` bloques y eso requerirá que el namenode guarde los metadatos para `750000` bloques para solo un archivo de 1GB. Dado que toda esta información relacionada con los metadatos se almacena en la memoria, se prefiere un tamaño de bloque mayor para guardar ese bit de carga adicional en el NameNode.

Ahora, nuevamente, el tamaño del bloque no se establece en un valor extremadamente alto como 1GB, etc. porque, idealmente, se inicia 1 mapeador para cada bloque de datos. Por lo tanto, si establece el tamaño de bloque en 1 GB, podría perder el paralelismo, lo que podría resultar en un rendimiento global más lento.

2.) **Tamaño dividido en HDFS** : Las divisiones en el procesamiento de Hadoop son los fragmentos lógicos de los datos. Cuando los archivos se dividen en bloques, hadoop no respeta ningún archivo bopundario. Simplemente divide los datos en función del tamaño del bloque. Diga que si tiene un archivo de 400 MB, con 4 líneas y cada línea con 100 MB de datos, obtendrá 3 bloques de `128 MB x 3` y `16 MB x 1` . Pero cuando se calculan las divisiones de entrada mientras se guardan los datos, los límites de archivo / registro se tienen en cuenta y en este caso tendremos 4 divisiones de entrada de 100 MB cada una, si está utilizando, por ejemplo, `NLineInputFormat` .

El tamaño dividido también se puede establecer por trabajo utilizando la propiedad `mapreduce.input.fileinputformat.split.maxsize`

Una muy buena explicación de Blocks vs Splits se puede encontrar en esta Respuesta SO /

Lea ¿Qué es HDFS? en línea: https://riptutorial.com/es/hadoop/topic/3845/-que-es-hdfs-

Capítulo 3: Comandos de Hadoop

Sintaxis

- Comandos de Hadoop v1: `hadoop fs -<command>`
- Comandos de Hadoop v2: `hdfs dfs -<command>`

Examples

Comandos Hadoop v1

1. Imprime la versión de Hadoop

```
hadoop version
```

2. Listar los contenidos del directorio raíz en HDFS

```
hadoop fs -ls /
```

3. Reportar la cantidad de espacio utilizado y disponible en el sistema de archivos montado actualmente

```
hadoop fs -df hdfs:/
```

4. Cuente el número de directorios, archivos y bytes bajo

las rutas que coinciden con el patrón de archivo especificado

```
hadoop fs -count hdfs:/
```

5. Ejecutar una utilidad de comprobación del sistema de archivos DFS

```
hadoop fsck - /
```

6. Ejecutar una utilidad de equilibrio de clúster

```
hadoop balancer
```

7. Cree un nuevo directorio llamado "hadoop" debajo del

/ user / training directory en HDFS. Ya que eres

actualmente iniciado sesión con el "entrenamiento" ID de usuario,

/ user / training es su directorio home en HDFS.

```
hadoop fs -mkdir /user/training/hadoop
```

8. Agregue un archivo de texto de muestra desde el directorio local

llamado "datos" al nuevo directorio que creó en HDFS

durante el paso anterior.

```
hadoop fs -put data/sample.txt /user/training/hadoop
```

9. Listar los contenidos de este nuevo directorio en HDFS.

```
hadoop fs -ls /user/training/hadoop
```

10. Agregue el directorio local completo llamado "minorista" al

/ user / training directory en HDFS.

```
hadoop fs -put data/retail /user/training/hadoop
```

11. Dado que / user / training es su directorio personal en HDFS,

Cualquier comando que no tenga una ruta absoluta es

---

interpretado como relativo a ese directorio. El siguiente

Por lo tanto, el comando listará su directorio de inicio, y

---

debe mostrar los elementos que acaba de agregar allí.

```
hadoop fs -ls
```

---

12. Ver cuánto espacio ocupa este directorio en HDFS.

```
hadoop fs -du -s -h hadoop/retail
```

---

13. Elimine un archivo 'clientes' del directorio "minorista".

```
hadoop fs -rm hadoop/retail/customers
```

---

14. Asegúrese de que este archivo ya no esté en HDFS.

```
hadoop fs -ls hadoop/retail/customers
```

15. Elimine todos los archivos del directorio "minorista" utilizando un comodín.

```
hadoop fs -rm hadoop/retail/*
```

16. Vaciar la basura.

```
hadoop fs -expunge
```

17. Finalmente, elimine todo el directorio minorista y todos

de sus contenidos en HDFS.

```
hadoop fs -rm -r hadoop/retail
```

18. Listar el directorio hadoop de nuevo

```
hadoop fs -ls hadoop
```

19. Agregue el archivo adquisiciones.txt desde el directorio local

nombrado "/ home / training /" en el directorio hadoop que creó en HDFS

```
hadoop fs -copyFromLocal /home/training/purchases.txt hadoop/
```

20. Para ver el contenido de tu archivo de texto adquisiciones.txt

que está presente en su directorio hadoop.

```
hadoop fs -cat hadoop/purchases.txt
```

21. Agregue el archivo adquisiciones.txt del directorio "hadoop" que está presente en el directorio HDFS

al directorio “datos” que está presente en su directorio local

```
hadoop fs -copyToLocal hadoop/purchases.txt /home/training/data
```

22. cp se utiliza para copiar archivos entre directorios presentes en HDFS

```
hadoop fs -cp /user/training/*.txt /user/training/hadoop
```

23. El comando '-get' se puede usar alternativamente al comando '-copyToLocal'

```
hadoop fs -get hadoop/sample.txt /home/training/
```

24. Muestra el último kilobyte del archivo "purchase.txt" a la salida estándar.

```
hadoop fs -tail hadoop/purchases.txt
```

25. Los permisos de archivo predeterminados son 666 en HDFS

Use el comando '-chmod' para cambiar los permisos de un archivo

```
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chmod 600 hadoop/purchases.txt
```

26. Los nombres predeterminados de propietario y grupo son entrenamiento, entrenamiento

Use '-chown' para cambiar el nombre del propietario y el nombre del grupo simultáneamente

```
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chown root:root hadoop/purchases.txt
```

27. El nombre predeterminado del grupo es entrenamiento.

Use el comando '-chgrp' para cambiar el nombre del grupo

```
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chgrp training hadoop/purchases.txt
```

28. Mover un directorio de una ubicación a otra

```
hadoop fs -mv hadoop apache_hadoop
```

29. El factor de replicación predeterminado para un archivo es 3.

Use el comando '-setrep' para cambiar el factor de replicación de un archivo

```
hadoop fs -setrep -w 2 apache_hadoop/sample.txt
```

30. Copie un directorio de un nodo en el cluster a otro

Utilice el comando '-distcp' para copiar,

Opción de sobrescritura para sobrescribir en archivos existentes.

-Actualizar el comando para sincronizar ambos directorios.

```
hadoop fs -distcp hdfs://namenodeA/apache_hadoop hdfs://namenodeB/hadoop
```

31. Comando para hacer que el nodo nombre salga del modo seguro

```
hadoop fs -expunge
sudo -u hdfs hdfs dfsadmin -safemode leave
```

32. Listar todos los comandos de shell del sistema de archivos hadoop

```
hadoop fs
```

33. Obtenga los valores de cuota de hdfs y el recuento actual de nombres y bytes en uso.

```
hadoop fs -count -q [-h] [-v] <directory>...<directory>
```

34. Por último, pero no menos importante, ¡siempre pide ayuda!

```
hadoop fs -help
```

Comandos Hadoop v2

appendToFile: anexar una fuente única o varias fuentes del sistema de archivos local al sistema de archivos de destino. También lee la entrada de la entrada estándar y se agrega al sistema de archivos de destino. Mantenga el que -

```
hdfs dfs -appendToFile [localfile1 localfile2 ..] [/HDFS/FILE/PATH..]
```

cat: copia las rutas de origen a la salida estándar.

```
hdfs dfs -cat URI [URI …]
```

chgrp: Cambia la asociación de grupo de archivos. Con -R, realiza el cambio de forma recursiva a través de la estructura del directorio. El usuario debe ser el propietario del archivo o el superusuario.

```
hdfs dfs -chgrp [-R] GROUP URI [URI …]
```

chmod: cambia los permisos de los archivos. Con -R, realiza el cambio de forma recursiva a través de la estructura del directorio. El usuario debe ser el propietario del archivo o el superusuario.

```
hdfs dfs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI …]
```

chown: Cambia el propietario de los archivos. Con -R, realiza el cambio de forma recursiva a través de la estructura del directorio. El usuario debe ser el superusuario.

```
hdfs dfs -chown [-R] [OWNER][:[GROUP]] URI [URI ]
```

copyFromLocal: funciona de manera similar al comando put, excepto que la fuente está restringida a una referencia de archivo local.

```
hdfs dfs -copyFromLocal <localsrc> URI
```

copyToLocal: funciona de manera similar al comando get, excepto que el destino está restringido a una referencia de archivo local.

```
hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
```

recuento: cuenta el número de directorios, archivos y bytes en las rutas que coinciden con el patrón de archivo especificado.

```
hdfs dfs -count [-q] [-h] <paths>
```

cp: copia uno o más archivos de un origen específico a un destino específico. Si especifica varias fuentes, el destino especificado debe ser un directorio.

```
hdfs dfs -cp URI [URI …] <dest>
```

du: muestra el tamaño del archivo especificado o el tamaño de los archivos y directorios que se encuentran en el directorio especificado. Si especifica la opción -s, muestra un resumen agregado de tamaños de archivo en lugar de tamaños de archivo individuales. Si especifica la opción -h, dé formato a los tamaños de archivo de forma "legible para los humanos".

```
hdfs dfs –du [–s] [–h] URI [URI …]
```

dus: muestra un resumen de los tamaños de archivo; equivalente a hdfs dfs -du –s.

```
hdfs dfs –dus <args>
```

expunge **:** Vacía la basura. Cuando elimina un archivo, no se elimina inmediatamente de HDFS, pero se le cambia el nombre a un archivo en el directorio / trash. Mientras el archivo permanezca allí, puede recuperarlo si cambia de opinión, aunque solo se puede restaurar la última copia del archivo eliminado.

```
hdfs dfs –expunge
```

obtener: copia los archivos al sistema de archivos local. Los archivos que fallan en una verificación de redundancia cíclica (CRC) aún se pueden copiar si especifica la opción -ignorecrc. El CRC es una técnica común para detectar errores de transmisión de datos. Los archivos de suma de comprobación CRC tienen la extensión .crc y se usan para verificar la integridad de los datos de otro archivo. Estos archivos se copian si especifica la opción -crc.

```
hdfs dfs –get [–ignorecrc] [–crc] <src> <localdst>
```

getmerge: Concatena los archivos en src y escribe el resultado en el archivo de destino local especificado. Para agregar un carácter de nueva línea al final de cada archivo, especifique la opción addnl.

```
hdfs dfs –getmerge <src> <localdst> [addnl]
```

ls: devuelve estadísticas para los archivos o directorios especificados.

```
hdfs dfs –ls <args>
```

lsr: sirve como la versión recursiva de ls; similar al comando de Unix ls -R.

```
hdfs dfs –lsr <args>
```

mkdir: crea directorios en una o más rutas especificadas. Su comportamiento es similar al comando mkdir -p de Unix, que crea todos los directorios que conducen al directorio especificado si aún no existen.

```
hdfs dfs –mkdir <paths>
```

moveFromLocal: Funciona de manera similar al comando put, excepto que la fuente se elimina después de que se copia.

```
hdfs dfs –moveFromLocal <localsrc> <dest>
```

mv: Mueve uno o más archivos de un origen específico a un destino específico. Si especifica varias fuentes, el destino especificado debe ser un directorio. No se permite mover archivos a través de sistemas de archivos.

```
hdfs dfs –mv URI [URI …] <dest>
```

put: copia los archivos del sistema de archivos local al sistema de archivos de destino. Este comando también puede leer la entrada de stdin y escribir en el sistema de archivos de destino.

```
hdfs dfs –put <localsrc> ... <dest>
```

rm: borra uno o más archivos especificados. Este comando no elimina directorios o archivos vacíos. Para omitir la papelera (si está habilitada) y eliminar los archivos especificados inmediatamente, especifique la opción -skipTrash.

```
hdfs dfs –rm [–skipTrash] URI [URI …]
```

rm r: Sirve como la versión recursiva de –rm.

```
hdfs dfs –rm –r [–skipTrash] URI [URI …]
```

setrep: cambia el factor de replicación para un archivo o directorio específico. Con -R, realiza el cambio de forma recursiva a través de la estructura del directorio.

```
hdfs dfs –setrep <rep> [–R] <path>
```

stat: muestra información sobre la ruta especificada.

```
hdfs dfs –stat URI [URI …]
```

cola: muestra el último kilobyte de un archivo especificado a la salida estándar. La sintaxis admite la opción Unix -f, que permite supervisar el archivo especificado. A medida que se agregan nuevas líneas al archivo mediante otro proceso, tail actualiza la pantalla.

```
hdfs dfs –tail [–f] URI
```

prueba: devuelve atributos del archivo o directorio especificado. Especifica -e para determinar si el archivo o directorio existe; -z para determinar si el archivo o directorio está vacío; y -d para determinar si el URI es un directorio.

```
hdfs dfs –test –[ezd] URI
```

texto: genera un archivo de origen especificado en formato de texto. Los formatos de archivo de entrada válidos son zip y TextRecordInputStream.

```
hdfs dfs –text <src>
```

touchz: crea un nuevo archivo vacío de tamaño 0 en la ruta especificada.

```
hdfs dfs –touchz <path>
```

Lea Comandos de Hadoop en línea: https://riptutorial.com/es/hadoop/topic/3870/comandos-de-hadoop

Capítulo 4: Datos de carga de Hadoop

Examples

Cargar datos en hadoop hdfs

PASO 1: CREAR UN DIRECTORIO EN HDFS, CARGAR UN ARCHIVO Y CONTENIDO DE LA LISTA

Aprendamos escribiendo la sintaxis. Podrá copiar y pegar los siguientes comandos de ejemplo en su terminal:

hadoop fs -mkdir:

Toma la ruta de URI como un argumento y crea un directorio o varios directorios.

Uso:

```
# hadoop fs -mkdir <paths>
```

Ejemplo:

```
hadoop fs -mkdir /user/hadoop
hadoop fs -mkdir /user/hadoop/dir1 /user/hadoop/dir2 /user/hadoop/dir3
```

hadoop fs -put:

Copia un solo archivo src o varios archivos src del sistema de archivos local al Sistema de archivos distribuidos de Hadoop.

Uso:

```
# hadoop fs -put <local-src> ... <HDFS_dest_path>
```

Ejemplo:

```
hadoop fs -put popularNames.txt /user/hadoop/dir1/popularNames.txt
```

hadoop fs -copyFromLocal:

Copia un solo archivo src o varios archivos src del sistema de archivos local al Sistema de archivos distribuidos de Hadoop.

Uso:

```
    # hadoop fs -copyFromLocal <local-src> ... <HDFS_dest_path>
```

Ejemplo:

```
    hadoop fs -copyFromLocal popularNames.txt /user/hadoop/dir1/popularNames.txt
```

hadoop fs -moveFromLocal:

Similar al comando put, excepto que el origen localsrc se elimina después de copiarlo.

Uso:

```
    # hadoop fs -moveFromLocal <local-src> ... <HDFS_dest_path>
```

Ejemplo:

```
    hadoop fs -moveFromLocal popularNames.txt /user/hadoop/dir1/popularNames.txt
```

HERRAMIENTA DE TRANSFERENCIA DE DATOS SQOOP:

También podemos cargar datos en HDFS directamente desde bases de datos relacionales usando Sqoop (una herramienta de línea de comandos para la transferencia de datos de RDBMS a HDFS y viceversa).

Uso:

```
$ sqoop import --connect CONNECTION_STRING --username USER_NAME --table TABLE_NAME
```

Ejemplo:

```
$ sqoop import --connect jdbc:mysql://localhost/db --username foo --table TEST
```

Lea Datos de carga de Hadoop en línea: https://riptutorial.com/es/hadoop/topic/3846/datos-de-carga-de-hadoop

Capítulo 5: Depuración del código Java de Hadoop MR en un entorno de desarrollo de eclipse local.

Introducción

Lo básico que se debe recordar aquí es que la depuración de un trabajo de Hadoop MR será similar a cualquier aplicación de depuración remota en Eclipse.

Un depurador o herramienta de depuración es un programa de computadora que se utiliza para probar y depurar otros programas (el programa "objetivo"). Es muy útil especialmente para un entorno Hadoop en el que hay poco espacio para el error y un pequeño error puede causar una gran pérdida.

Observaciones

Eso es todo lo que necesitas hacer.

Examples

Pasos para la configuracion

Como sabría, Hadoop se puede ejecutar en el entorno local en 3 modos diferentes:

1. Modo local
2. Modo Pseudo Distribuido
3. Modo totalmente distribuido (Cluster)

Por lo general, ejecutará la configuración local de hadoop en modo pseudo-distribuido para aprovechar HDFS y Map Reduce (MR). Sin embargo, no puede depurar programas de MR en este modo, ya que cada tarea de Map / Reduce se ejecutará en un proceso JVM separado, por lo que debe volver al modo Local, donde puede ejecutar sus programas de MR en un solo proceso JVM.

Estos son los pasos rápidos y simples para depurar esto en su entorno local:

1. Ejecute hadoop en modo local para la depuración, de modo que las tareas del asignador y del reductor se ejecuten en una única JVM en lugar de JVM separadas. Los pasos a continuación te ayudan a hacerlo.

2. Configure HADOOP_OPTS para habilitar la depuración de modo que cuando ejecute su trabajo Hadoop, estará esperando a que el depurador se conecte. A continuación se muestra el comando para depurar lo mismo en el puerto 8080.

(exportar HADOOP_OPTS = ”- agentlib: jdwp = transporte = dt_socket, servidor = y, suspender = y, dirección = 8008“)

3. Configure el valor fs.default.name en core-site.xml en el archivo: /// de hdfs: //. No utilizarás hdfs en modo local.

4. Configure el valor mapred.job.tracker en mapred-site.xml en local. Esto le indicará a Hadoop que ejecute tareas de MR en una única JVM.

5. Cree la configuración de depuración para Eclipse y establezca el puerto en 8008 - cosas típicas. Para eso, vaya a las configuraciones del depurador y cree un nuevo tipo de configuración de la Aplicación Java Remota y establezca el puerto como 8080 en la configuración.

6. Ejecute su trabajo hadoop (estará esperando a que se conecte el depurador) y luego inicie Eclipse en modo de depuración con la configuración anterior. Asegúrese de poner primero un punto de quiebre.

Lea Depuración del código Java de Hadoop MR en un entorno de desarrollo de eclipse local. en línea: https://riptutorial.com/es/hadoop/topic/10063/depuracion-del-codigo-java-de-hadoop-mr-en-un-entorno-de-desarrollo-de-eclipse-local-

Capítulo 6: Introducción a MapReduce

Sintaxis

- Para ejecutar el ejemplo, la sintaxis del comando es:

```
bin/hadoop jar hadoop-*-examples.jar wordcount [-m <#maps>] [-r <#reducers>] <in-dir>
<out-dir>
```

- Para copiar datos en HDFS (desde local):

```
bin/hadoop dfs -mkdir <hdfs-dir> //not required in hadoop 0.17.2 and later
bin/hadoop dfs -copyFromLocal <local-dir> <hdfs-dir>
```

Observaciones

Programa Word Count utilizando MapReduce en Hadoop.

Examples

Programa de conteo de palabras (en Java y Python)

El programa de conteo de palabras es como el programa "Hello World" en MapReduce.

Hadoop MapReduce es un marco de software para escribir aplicaciones que procesan vastas cantidades de datos (conjuntos de datos de varios terabytes) en paralelo en grandes clusters (miles de nodos) de hardware básico de una manera confiable y tolerante a fallos.

Un trabajo MapReduce generalmente divide el conjunto de datos de entrada en fragmentos independientes que son procesados por las tareas del mapa de una manera completamente paralela. El marco ordena los resultados de los mapas, que luego se ingresan en las tareas de reducción. Normalmente, tanto la entrada como la salida del trabajo se almacenan en un sistema de archivos. El marco se encarga de programar las tareas, monitorearlas y volver a ejecutar las tareas fallidas.

Ejemplo de conteo de palabras:

El ejemplo de WordCount lee archivos de texto y cuenta la frecuencia con la que aparecen las palabras. La entrada son archivos de texto y la salida son archivos de texto, cada línea de los cuales contiene una palabra y el conteo de la frecuencia con la que se produjo, separados por una pestaña.

Cada mapeador toma una línea como entrada y la divide en palabras. A continuación, emite un par de clave / valor de la palabra y cada reductor suma los conteos de cada palabra y emite una única clave / valor con la palabra y la suma.

Como optimización, el reductor también se utiliza como combinador en las salidas del mapa. Esto reduce la cantidad de datos enviados a través de la red al combinar cada palabra en un solo registro.

Código de conteo de palabras:

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
 }

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);
```

```
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
 }

}
```

Para ejecutar el ejemplo, la sintaxis del comando es:

```
bin/hadoop jar hadoop-*-examples.jar wordcount [-m <#maps>] [-r <#reducers>] <in-dir> <out-
dir>
```

Se leen todos los archivos en el directorio de entrada (llamado in-dir en la línea de comando arriba) y los conteos de palabras en la entrada se escriben en el directorio de salida (llamado out-dir arriba). Se supone que tanto las entradas como las salidas se almacenan en HDFS. Si su entrada no está ya en HDFS, sino que se encuentra en un sistema de archivos local en algún lugar, debe copiar los datos en HDFS utilizando un comando como este:

```
bin/hadoop dfs -mkdir <hdfs-dir> //not required in hadoop 0.17.2 and later
bin/hadoop dfs -copyFromLocal <local-dir> <hdfs-dir>
```

Ejemplo de Word Count en Python:

mapper.py

```
import sys
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        print '%s\t%s' % (word, 1)
```

reducer.py

```
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    # remove leading and trailing whitespaces
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
```

```
        # count was not a number, so silently
        # ignore/discard this line
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

El programa anterior se puede ejecutar usando `cat filename.txt | python mapper.py | sort -k1,1 | python reducer.py`

Lea Introducción a MapReduce en línea: https://riptutorial.com/es/hadoop/topic/3879/introduccion-a-mapreduce

Capítulo 7: matiz

Introducción

Hue es una interfaz de usuario para conectarse y trabajar con la mayoría de las tecnologías de Bigdata utilizadas comúnmente como HDFS, Hive, Spark, Hbase, Sqoop, Impala, Pig, Oozie, etc. Hue también admite la ejecución de consultas en bases de datos relacionales.

Hue, una aplicación web de django, se creó principalmente como un banco de trabajo para ejecutar consultas de Hive. Más tarde, la funcionalidad de Hue se incrementó para admitir diferentes componentes del ecosistema Hadoop. Está disponible como software de código abierto bajo la Licencia Apache.

Examples

Proceso de configuración

Dependencias de instalacion

Los detalles del proceso de instalación de Hue no están disponibles para la mayoría de los sistemas operativos, por lo que, dependiendo del sistema operativo, puede haber variaciones en las dependencias que necesita instalar antes de ejecutar el script de instalación que se incluye en el paquete de instalación :

CentOS

```
sudo yum install ant
sudo yum install python-devel.x86_64
sudo yum install krb5-devel.x86_64
sudo yum install krb5-libs.x86_64
sudo yum install libxml2.x86_64
sudo yum install python-lxml.x86_64
sudo yum install libxslt-devel.x86_64
sudo yum install mysql-devel.x86_64
sudo yum install openssl-devel.x86_64
sudo yum install libgsasl-devel.x86_64
sudo yum install sqlite-devel.x86_64
sudo yum install openldap-devel.x86_64
sudo yum install -y libffi libffi-devel
sudo yum install mysql-devel gcc gcc-devel python-devel
sudo yum install rsync
sudo yum install maven
wget https://bootstrap.pypa.io/ez_setup.py -O - | sudo python
```

1. GMP

- CentOS> 7.x
  ```
  sudo yum install libgmp3-dev
  ```

- CentOS <6.x
  `sudo yum install gmp gmp-devel gmp-status`

Instalación de Hue en Ubuntu

Esta instalación asume que `hadoop` está preinstalado bajo el usuario `hadoop` .

Requisitos previos:

Hue depende de estos siguientes paquetes

1. gcc
2. g ++
3. libxml2-dev
4. libxlst-dev
5. libsasl2-dev
6. libsasl2-modules-gssapi-mit
7. libmysqlclient-dev
8. python-dev
9. herramientas de configuración de python
10. libsqlite3-dev
11. hormiga
12. libkrb5-dev
13. libtidy-0.99-0
14. libldap2-dev
15. libssl-dev
16. libgmp3-dev

Instalando todos los paquetes

```
sudo apt-get update
sudo apt-get install gcc g++ libxml2-dev libxslt-dev libsasl2-dev libsasl2-modules-gssapi-mit
libmysqlclient-dev python-dev python-setuptools libsqlite3-dev ant libkrb5-dev libtidy-0.99-0
libldap2-dev libssl-dev libgmp3-dev
```

Instalacion y configuracion

Realizando la instalación como usuario de `hadoop` .

```
su - hadoop
```

1. Descarga Hue de gethue.com (este enlace es un ejemplo obtenido del sitio web de Hue)

   ```
   wget https://dl.dropboxusercontent.com/u/730827/hue/releases/3.9.0/hue-3.9.0.tgz
   ```

2. Extraer el tarball descargado

   ```
   tar -xvf hue-3.9.0.tgz
   ```

3. Ejecutar comando de instalación

```
cd hue-3.9.0
PREFIX=/home/hadoop/ make install
```

4. Una vez completado el proceso anterior,

   Actualizar el archivo `~/.bashrc` ,

   ```
   export HUE_HOME=/home/hadoop/hue
   export PATH=$PATH:$HUE_HOME/build/env/bin
   ```

   fuente después de agregar las entradas, fuente ~ / .bashrc

5. Configurar Hue (3 archivos para editar)

```
cd $HUE_HOME/desktop/conf
```

- hue.ini

```
[desktop]
 server_user=hadoop
 server_group=hadoop
 default_user=hadoop
 default_hdfs_superuser=hadoop
```

```
cd $HADOOP_CONF_DIR
```

- core-site.xml

```
<property>
    <name>hadoop.proxyuser.hadoop.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.hadoop.groups</name>
    <value>*</value>
</property>
```

- hdfs-site.xml

```
<property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
</property>
```

6. Iniciar Hue (iniciar demonios Hadoop si aún no ha comenzado)

   ```
   nohup supervisor &
   ```

7. Inicie sesión en la interfaz web de Hue: http: // localhost: 8888

   nombre de usuario: hadoop

   contraseña : user_choice

Lea matiz en línea: https://riptutorial.com/es/hadoop/topic/6133/matiz

Creditos

S. No	Capítulos	Contributors
1	Empezando con hadoop	Ani Menon, Community, franklinsijo, Harinder, ItayB, Sandeep Chatterjee, Shailesh Kumar Dayananda, sunkuet02, Udeet Solanki, Venkata Karthik
2	¿Qué es HDFS?	Ani Menon, NeoWelkin, neuromouse, philantrovert, Suraj Kumar Yadav, Tejus Prasad
3	Comandos de Hadoop	Ambrish, Ani Menon, jedijs, philantrovert
4	Datos de carga de Hadoop	Ani Menon, Backtrack, BruceWayne, NeoWelkin, Tejus Prasad
5	Depuración del código Java de Hadoop MR en un entorno de desarrollo de eclipse local.	Manish Verma
6	Introducción a MapReduce	Ani Menon, Arduino_Sentinel, Tejus Prasad, Udeet Solanki, user3335966
7	matiz	andriosr, franklinsijo